In some embodiment voting databases **140** can correspond to some or all of databases **150** from FIG. **1**. In some embodiments voting databases **1404** can comprise two separate databases votes database **1405** and signature database **1406**. In some embodiments, the blockchain access layer sends data about the actual votes that the user submitted to votes database **1405** and submits data about the identity of the user including the digitized signature and VoterID to signature database **1406** (shown as **1***a* on FIG. **14**). Separating the storage of the identity of the voter from the votes case helps to ensure that the votes are anonymous. In some embodiments, the data in both voting databases can be encrypted through the use of an unbound key pair stored in in a unbound key pair cache or database. In some embodiments, the unbound key pair cache or databases stores multiple keys in separate key modules and then encrypts the data using each portion of the key in each module. In some embodiments, this can be two or three modules. This increases security by requiring that an attacker compromise all the modules in order to decrypt the data. In some embodiments, the unbound key pair cache or database can correspond to key store **604**.

[0123] In some embodiments, the blockchain abstraction layer **1402** can also create an entry on the submitted vote blockchain **1407** as it creates the entries in the voting databases **140** (shown as **1***b* on FIG. **14**). The blockchain abstraction layer **1402** records, on the submitted vote blockchain **1407** to stores information about the voting. In some embodiments, for each ballot submitted, the blockchain abstraction layer **1402** creates a voteID, a unique entry on the submitted vote blockchain **1407** that contains a unique number that corresponds to the cast ballot or an instance of the vote or of a receipt of ballot selections, and a pointer that points to the vote record stored in the votes database **1405**, a pointer that points to the data in the signature database **1406**, a hash of the digitized signature of the voter, and a count of all of the votes currently submitted.

[0124] Once the data has been stored in the voting databases **1404**, and the above data has been written to the submitted vote blockchain **1407** the blockchain abstraction layer **1402** can also transmit the signature data to a vote by mail election official application **1403**. The application **1403** can be used by an election official to verify that the correct voter casts the votes. In some embodiments, this can be done by comparing the digitized signature in the signature data with the signature on file when the voter registered to vote. An election official can perform this comparison to validate a voter and to approve the casting of votes by the voter. The election official uses application **1403** to inform the blockchain abstraction layer **1402** that the vote is approved (shown as **2** on FIG. **14**).

[0125] In some embodiments, once the voter is approved, the blockchain abstraction layer **1402** creates an entry on accepted vote blockchain **1408**. The accepted vote blockchain **1408** can be stored in blockchain database, or as part of a blockchain distributed ledger, or on another desired blockchain architecture. In some embodiments, accepted vote blockchain **1408** contains the VoteID, and includes the actual ballot choices for the vote, and the tabulation of all the votes currently casted in the election. In some embodiments, once this entry is created on the accepted vote blockchain **1408** all links between the actual votes cast and the affidavit or identity of the voter casting the votes are deleted.

[0126] In some embodiments, the votes can also be verified using verification contract database **1409** as discussed more below.

[0127] FIG. **15** displays one embodiment of a system that can be used to verify data sent out of the secure voting system using a verification smart contract. The system of verifications described herein provides auditability and a strong reporting mechanism. In some embodiments, this system can be combined with any and all features of the systems described elsewhere herein in order to create a secure voting system. In some embodiments, the system comprises a vote by mail import application **1501**. Vote by mail import application **1501** is an application that is used to transmit or import election data **1502** into the blockchain abstraction layer **1402**. In some embodiments, election data can be any data that is transmitted to blockchain abstraction layer **1402** from any other part of the system, including information creating or establishing the ballot, any vote data sent by the VBM application **1401**, or any data sent by election official application **1403**.

[0128] In some embodiments, when the blockchain access layer **1401** receives election data **1502**, the blockchain abstraction layer **1402** can use the verification service module **1504** to create a hash of the data that was transmitted. In some embodiments, the verification service module **1504** can be part of the blockchain abstraction layer **1402**. In other embodiments, the verification service module **1504** can be a separate component. This hash is then stored on the verification contract database **1409** as part of a blockchain. In some embodiments, the hash can also be transmitted to neutral third party location to provide an additional level of security for the system. At the same time, the blockchain abstraction layer **1402** stores the data in the database **1503**. In some embodiments, the data is stored in any of the databases **150** described above or in the voting databases **1404**, or any other database that the blockchain abstraction layer **1402** can use to store data.

[0129] In some embodiments, when any client device, such as applications **1401** and **1403**, then reads data from blockchain abstraction layer **1402**, the blockchain abstraction layer **1402** can read the data from the database **1503** and also retrieve the hash from verification contract database **1409** and transmit the data to the client device. Then the client device can calculate its own hash of the transmitted data and compare it the hash of the data it received to determine if the data has been altered. If the calculated hash and the received hash do not match, a flag or error can be generated to indicate that data has been altered, corrupted, tampered with, or can identify another problem.

[0130] In some embodiment, the hash comparison can occur whenever data is retrieved by the system. In other embodiments, the system will only calculate and compare hashes at certain checkpoints. These checkpoints can occur at the following points: data ingestion, when the election data is ingested; Ballot storage, when ballot contents are stored to ensure they are unchanged from when they were received; ballot presentment to voter, such as checking the validity or integrity of the ballot before it is provided to the voter; vote submission, when the votes are submitted; vote tabulation, when the votes are counted to ensure that only votes that have not been tampered with are counted; vote metrics, to ensure that the auditability metrics are secure. For example, the system can periodically check the hash of the ballot information to ensure that the ballot contents did